NOSC TR 1152

AD-A178 382

NOSC

NAVAL OCEAN SYSTEMS CENTER San Diego, California 92152-5000

(2)

Technical Report 1152
January 1987

An Altitude-Error Display for Height-Finder Radar

A. E. Barrios

"Original contains color plates: All DTIC reproductions will be in black and white"

Approved for public release, distribution is unlimited

87 3 24 0

NAVAL OCEAN SYSTEMS CENTER

San Diego, California 92152-5000

E. G. SCHWEIZER, CAPT, USN
Commander

R. M. HILLYER
Technical Director

ADMINISTRATIVE INFORMATION

The work described in this report was performed by the Tropospheric Branch (Code 543), Naval Oceans Systems Center, for the Office of Naval Technology, Office of Chief of Naval Research.

Released by
H.V. Hitney, Head
Tropospheric Branch

Under authority of
J.H. Richter, Head
Ocean and Atmospheric
Sciences Division

JA

PAGES______ ARE MISSING IN ORIGINAL DOCUMENT

AD-A178382

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S): NOSC TR 1152	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: Naval Ocean Systems Center; 6b. OFFICE SYMBOL (if applicable): Code 543	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code): San Diego, CA 92152-5000	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: Office of Naval Technology; 8b. OFFICE SYMBOL (if applicable): ONT	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code): Office of Chief of Naval Research, Washington, DC 22217	10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	AGENCY ACCESSION NO.
62759N	N02C	RW59	DN888 715

11. TITLE (include Security Classification)
An Altitude-Error Display for Height-Finder Radar

12. PERSONAL AUTHOR(S)
A.E. Barrios

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM Sep 85 TO Apr 86	January 1987	51

16. SUPPLEMENTARY NOTATION
[illegible] include!

17. COSATI CODES

FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Height-finder radars, Target height, Ray-trace display, Altitude-error display, HP9000-500 series computer, and IREPS

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Strong ducting conditions affect height-finder radars in giving accurate target positions. An altitude-error display has been developed to show the amount of error present.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: [X] UNCLASSIFIED/UNLIMITED [ ] SAME AS RPT [ ] DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION: UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL: A.E. Barrios	22b. TELEPHONE (Include Area Code): (619) 225-7247; 22c. OFFICE SYMBOL: Code 543

DD FORM 1473, 84 JAN

83 APR EDITION MAY BE USED UNTIL EXHAUSTED
ALL OTHER EDITIONS ARE OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

SUMMARY

An altitude-error display for height-finder radar has been developed on the HP9000 – 500-series computer. The display looks like an ordinary ray-trace display except the color of the rays are dependent on the height difference, as compared to a standard atmosphere for the same elevation angle and range.

CONCLUSIONS

Since this effort only considered alternate displays of well-established ray-tracing theory, no attempt was made to validate the accuracy of any of the resulting displays.

RECOMMENDATION

The altitude-error displays reported here should be incorporated into the coverage diagram of the Integrated Refractive Effects Prediction System (IREPS).

CONTENTS

INTRODUCTION

Strong ducting conditions occur over many ocean areas. These conditions affect height-finder radars (such as the SPS-48) in giving accurate target positions. Most height-finder radars calculate altitude based on a standard atmosphere. When ducting conditions are present, large errors can occur between the calculated and the true target height, depending on the transmitter height and the target range. In many cases, these errors are greater than 50 percent. A height-finder radar altitude-error display has been developed to show the amount of error present. The display looks like an ordinary ray-trace display, except the color of the rays are dependent on the height difference, as compared to a standard atmosphere for the same elevation angle and range.

DD-1473

BACKGROUND

The atmosphere is considered to consist of vertical layers, each with a certain gradient of the index of refraction. The layers are assumed to be horizontally homogeneous. Each profile consists of at least two layers. Each layer is associated with a height, H(i), and modified refractivity or M-unit value, M(i). Modified refractivity, M, is related to the index of refraction, n, by

$$M = \left[n - 1 + \frac{z}{a}\right] \cdot 10^6$$

where a is the mean earth's radius, and z is the height above the earth's surface.

A ray path under standard atmospheric conditions will bend downward at a rate less than the curvature of the earth, so to an observer stationed on the earth's surface, the ray will appear to bend upward. A trapped ray is one that, because of a trapping layer, will bend downward at a rate exceeding the curvature of the earth. A trapping layer can be very easily identified by a negative M-gradient. Examples of profiles that represent a surface-based duct and an elevated duct are shown in figure 1. Two other types of refraction that describe the relation between modified refractivity and height are subrefraction and superrefraction. A subrefractive profile will cause rays to be bent less than the normal or standard, while a superrefractive profile bends rays at a rate exceeding the normal but not enough to cause trapping.

The gradient, Dmdh, is the change in M-units with respect to the change in height and is defined as

$$Dmdh(i) = \left[\frac{M(i+1) - M(i)}{H(i+1) - H(i)}\right] \cdot 10^{-3}$$

Table 1 shows the relation between the M-gradient and the different types of refraction. Figure 2 gives a clear picture of the relative bending among the different types.

An individual ray trace begins with an elevation angle specified at the source height, Ht, and consists of a series of calculations to determine either height at a specified range or range at a specified height. All calculations can be described by one of the following six cases. The variables are defined as

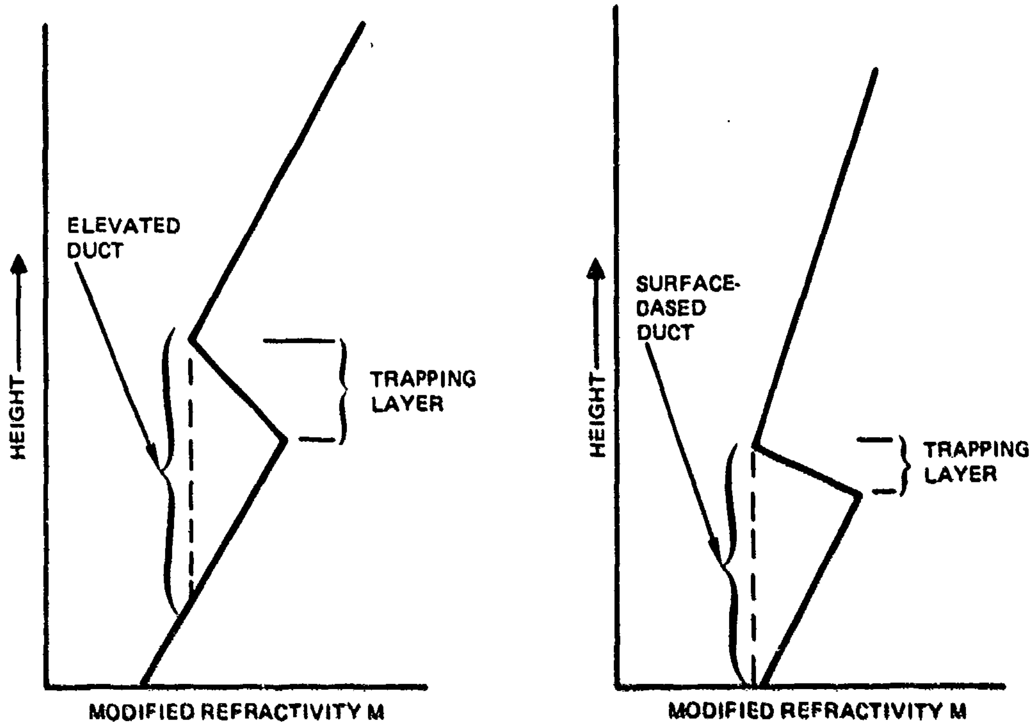


Figure 1. Examples of elevated and surface-based ducts from elevated layers.

Table 1. The relation between M-gradient and the different types of refraction.

Types of Refraction	M-Gradient
Trapping	< = 0 M/km
	< = 0 M/kft
Superrefractive	0 to 79 M/km
	0 to 24 M/kft
Standard	79 to 157 M/km
	24 to 48 M/kft
Subrefractive	>157 M/km
	48 M/kft

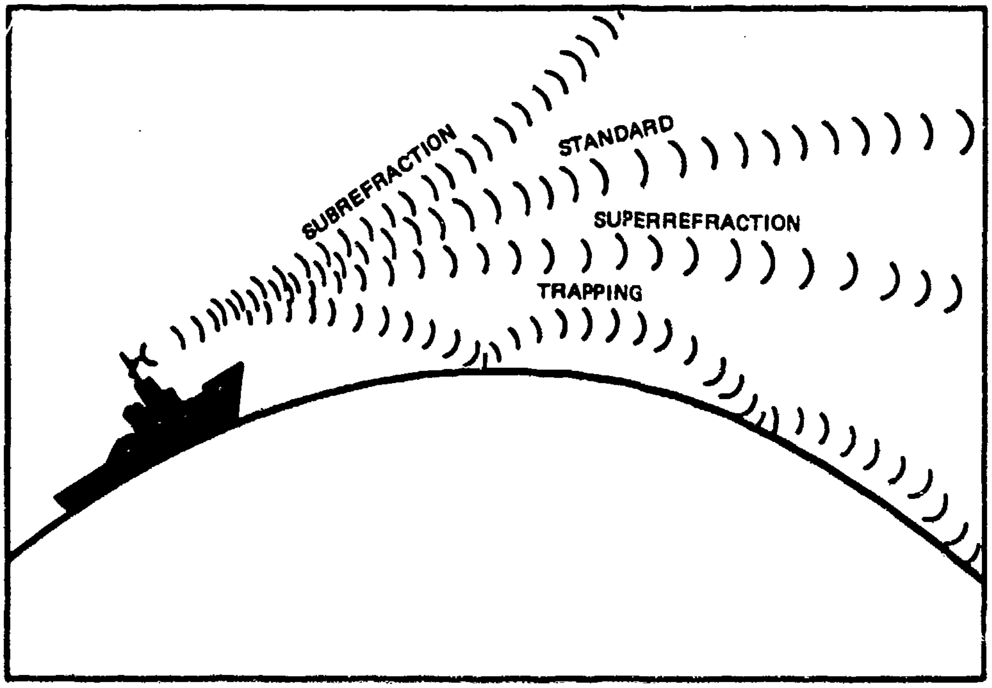


Figure 2. Relative bending for the four types of refraction.

α = elevation angle at beginning of calculation (rad)

α' = elevation angle at end of calculation (rad)

h = height at beginning of calculation (m)

h' = height at end of calculation (m)

r = range at beginning of calculation (km)

r' = range at end of calculation (km)

Case 1: $\alpha > 0$, h' known (figure 3)

$$\alpha' = \sqrt{\alpha^2 + 2 \cdot 10^{-3}\, \mathrm{Dmdh}(i)\,(h' - h)} \quad , \quad \text{if } H(i) \leq h < h' \leq H(i+1)$$

$$r' = r + \frac{\alpha' - \alpha}{\mathrm{Dmdh}(i)}$$

If the source is in a duct, trapping may occur for some initial elevation angles. In this case, the radicand for α' becomes negative, and h' reaches a maximum height. For this example,

$$\alpha' = 0 \quad ,$$

$$r' = r - \frac{\alpha}{\mathrm{Dmdh}(i)} \quad , \text{and}$$

$$h' = h - \frac{\alpha^2}{2 \cdot 10^{-3}\, \mathrm{Dmdh}(i)}$$

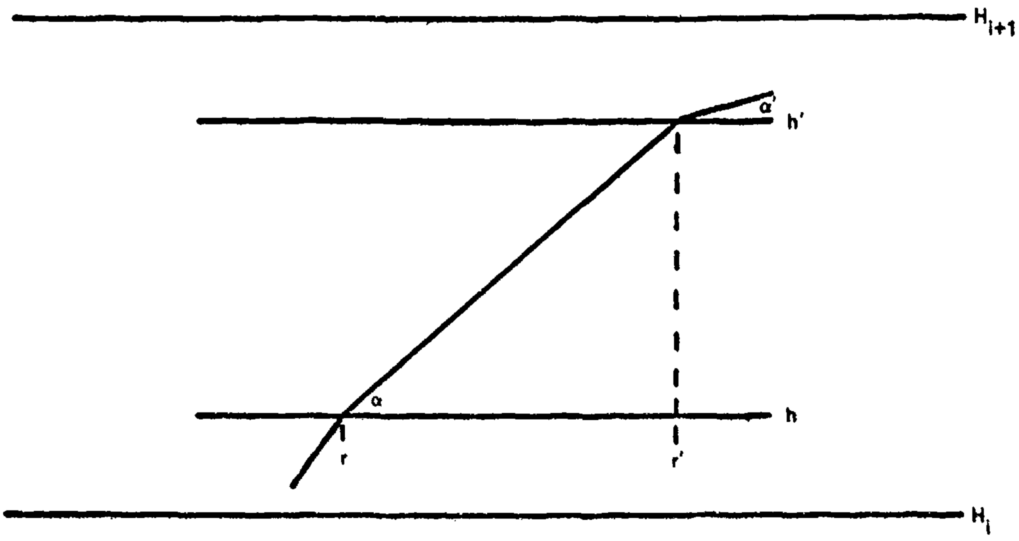


Figure 3. Ray trace variables for $\alpha > 0$.

Case 2: $\alpha > 0$, r' known (figure 3 applies)

$$\alpha' = \alpha + \mathrm{Dmdh}(i)\,(r' - r) \quad , \quad \text{if } h < H(i+1),$$

$$h' = h + \frac{\alpha'^2 - \alpha^2}{2 \cdot 10^{-3}\,\mathrm{Dmdh}(i)}$$

Case 3: $\alpha < 0$, h' known (figure 4)

$$\alpha' = -\sqrt{\alpha^2 + 2 \cdot 10^{-3}\,\mathrm{Dmdh}(i)\,(h' - h)} \quad , \quad \text{if } H(i+1) \geq h > h' \geq H(i),$$

$$r' = r + \frac{\alpha' - \alpha}{\mathrm{Dmdh}(i)}$$

A ray that is initially downgoing may eventually become upgoing as α increases. The ray reaches a minimum height, and in this case, the radicand for α' becomes negative. Therefore,

$$\alpha' = 0 \quad ,$$

$$r' = r - \frac{\alpha}{\mathrm{Dmdh}(i)} \quad ,$$

$$h' = h - \frac{\alpha^2}{2 \cdot 10^{-3}\,\mathrm{Dmdh}(i)} \quad .$$

Case 4: $\alpha < 0$, r' known (figure 4 applies)

$$\alpha' = \alpha + \mathrm{Dmdh}(i)\,(r' - r) \quad , \quad \text{if } h' > H(i).$$

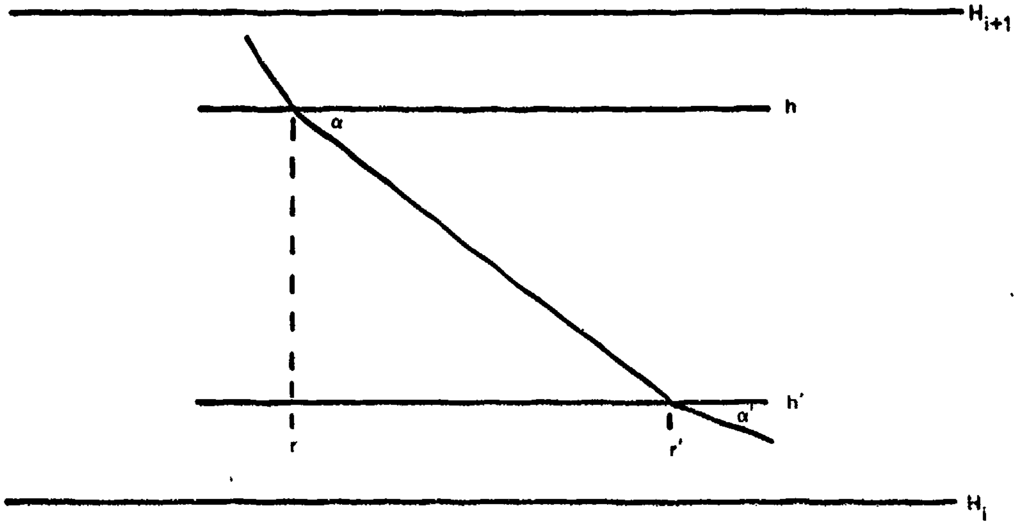


Figure 4. Ray trace variables for $\alpha < 0$.

$$h' = h + \frac{\alpha'^2 - \alpha^2}{2 \cdot 10^{-3}\,\mathrm{Dmdh(i)}}$$

Case 5: $\alpha = 0$

If Dmdh(i) $>$ 0, use Case 1 or Case 2 as appropriate.

If Dmdh(i) $<$ 0, use Case 3 or Case 4 as appropriate.

Case 6: A ray launched at the inflection point (figure 5) with $\alpha = 0$ will stay at that height.

PROGRAM

The program plots height vs. range for each ray. Reflected rays are not traced. Each ray is drawn in several colors, depending on the error scale chosen by the user.

Initially, there are four options available to the user: (1) delete data file, (2) add data file, (3) edit data file, and (4) run the program for a display. If option 4 is picked, a list of available data files will appear on the screen. After a data file has been chosen, there are several parameters the user must enter into the program:

1. Maximum height for the display in feet or meters.
2. Maximum range for the display in nautical miles or kilometers.
3. Antenna height in feet or meters.
4. Lower elevation angle. Lower angular limit in milliradians.
5. Upper elevation angle. Upper angular limit in milliradians.
6. Number of rays. This is the number of rays to be traced.

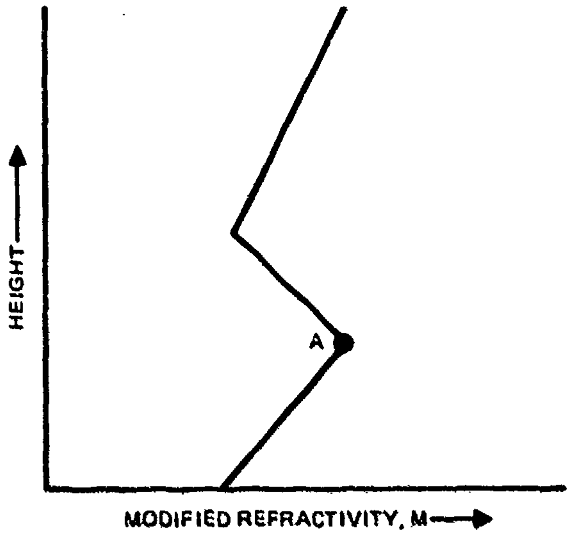


Figure 5. Breakpoint at A.

7. Error scale. The height-error scale can be an absolute- or percent-error scale, indicated by entering "A" or "P". The default is "A".

8. Error increment. This value determines the scale by which the colors are defined. If using an absolute-error scale, the error increment is entered in feet or meters.

Normal output for the display is the screen, but a hardcopy print can be selected.

Once the data has been read from the selected file, the program performs a linear extrapolation to find the M-unit value at the surface and at the maximum plot height, if necessary. These values are added to the height and M-unit arrays and the gradients (Dmdh) are then calculated.* The initial launch angles, calculated from the lower elevation angle, the upper elevation angle, and the number of rays, are also put into an array.

The ray trace is performed by range increments (1/50th of the maximum plot range) i.e., beginning with the initial launch angle and antenna height, a new angle and height are calculated to a specified range, plotted, the range is incremented, the next angle and height are calculated, plotted, and so forth until the maximum height or maximum range has been reached. As each new angle is calculated, the program will branch to the appropriate case (as discussed in the previous section). With each new height calculated the program will trace a ray to the same range using the same initial launch angle for a standard atmosphere. The difference between the two heights gives the height error, and the ray will be drawn in the appropriate color.

If the antenna height is at a breakpoint where Dmdh is positive beneath the point and negative above, the program will calculate through an infinite number of maximums and minimums for sufficiently small initial launch angles, resulting in a complete halt of the program. Also, because of the dot-matrix screen, some height increments may be too small to be noticed in plotting, and what the program calculates as maximums and minimums may look like a straight line across the screen. Therefore, a restriction on the initial launch angles is made. Depending on the angular limits used, the program will not allow traces of

*To avoid Dmdh approaching zero, a value of 10e-6 was assigned to Dmdh if it fell in the region -10e-6 < Dmdh < 10e-6.

rays launched within a certain angle above and below the horizontal. A single horizontal ray trajectory is drawn for launch angles within this range and the next greater angle in the launch angle array is selected upon completion of this ray path.

SAMPLE DISPLAYS

Figure 6 shows a height-finder radar altitude-error display for an elevated duct. Figure 7 shows two ray-trace displays, one for a standard atmosphere (black rays) and one for the same elevated duct (red rays). The height difference calculated for a specific range between the two sets of rays is shown. The middle ray, launched at zero radians, is refracted the most and therefore produces the most error. This can also be seen in figure 6 where the red area (for the shortest range) is associated with rays launched near zero radians.

The program does not distinguish between height difference above or below the standard. For instance, at a height of 10,000 feet and a range of 100 miles, one may get an error of 1000 feet, which could be 1000 feet above the standard or 1000 feet below the standard. As can be seen in figure 8, the ray at -10 mrad bends above the standard then crosses over to bend below (rays in standard atmosphere are in black). Again, the middle ray at zero radians gives the highest error. The error display is shown in figure 9. The error display for the same profile using 200 rays is shown in figure 10. In displays such as this, one can see rays increasing in height error (as well as height and range), then starting to decrease in error. Figure 8 shows why this is so.

Another example of this type of display is shown in figure 11 with the corresponding ray trace shown in figure 12. In figure 11, the first ray is reflected and therefore, not traced. From the third ray on, the rays bend toward the standard to produce a decrease in height error. The "full" error display (using 200 rays) is shown in figure 13.

Although the error displays are meant to be used for surface height-finder radars, figures 6 and 10 are displays for airborne height-finder radars and are shown only for demonstration purposes.

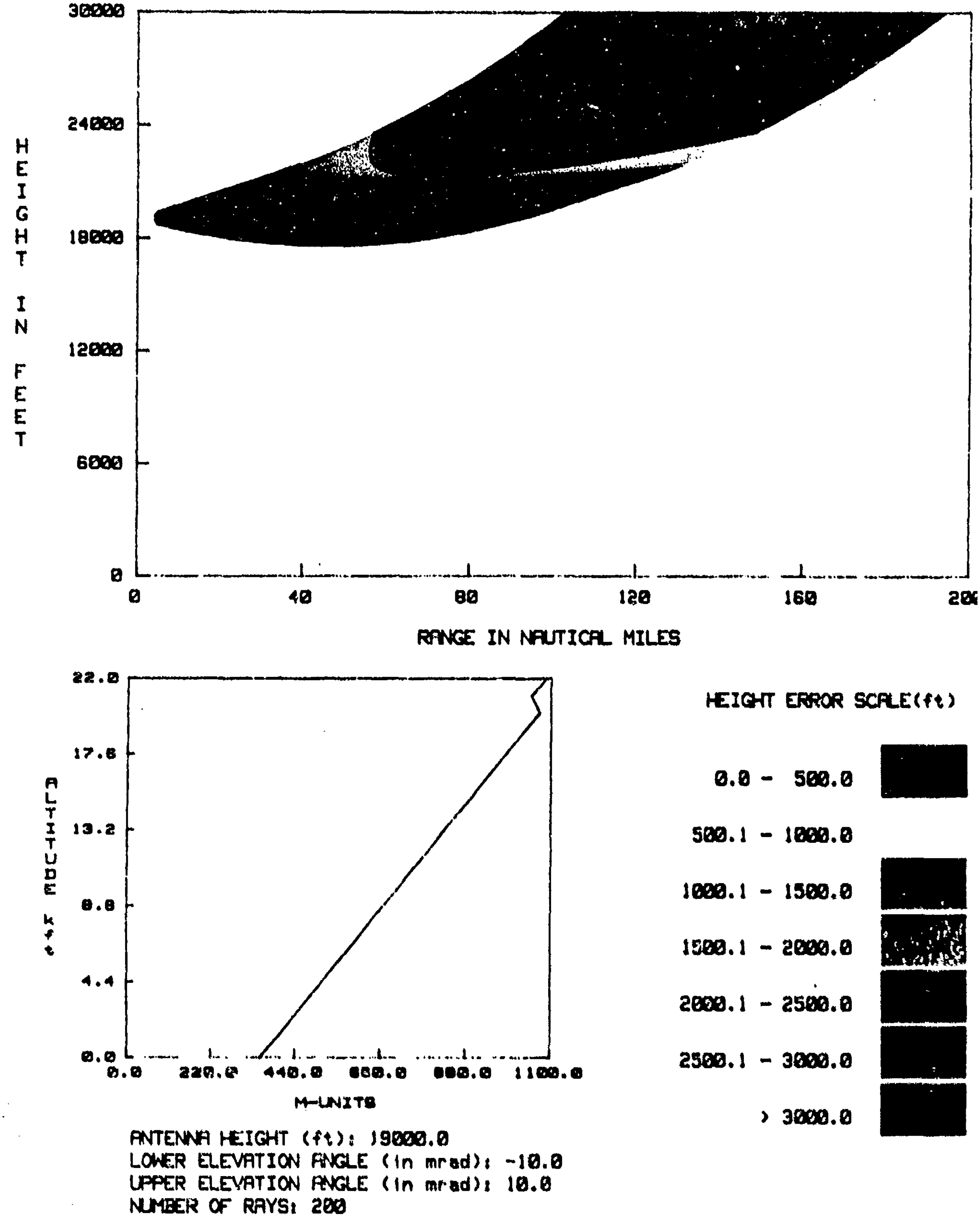
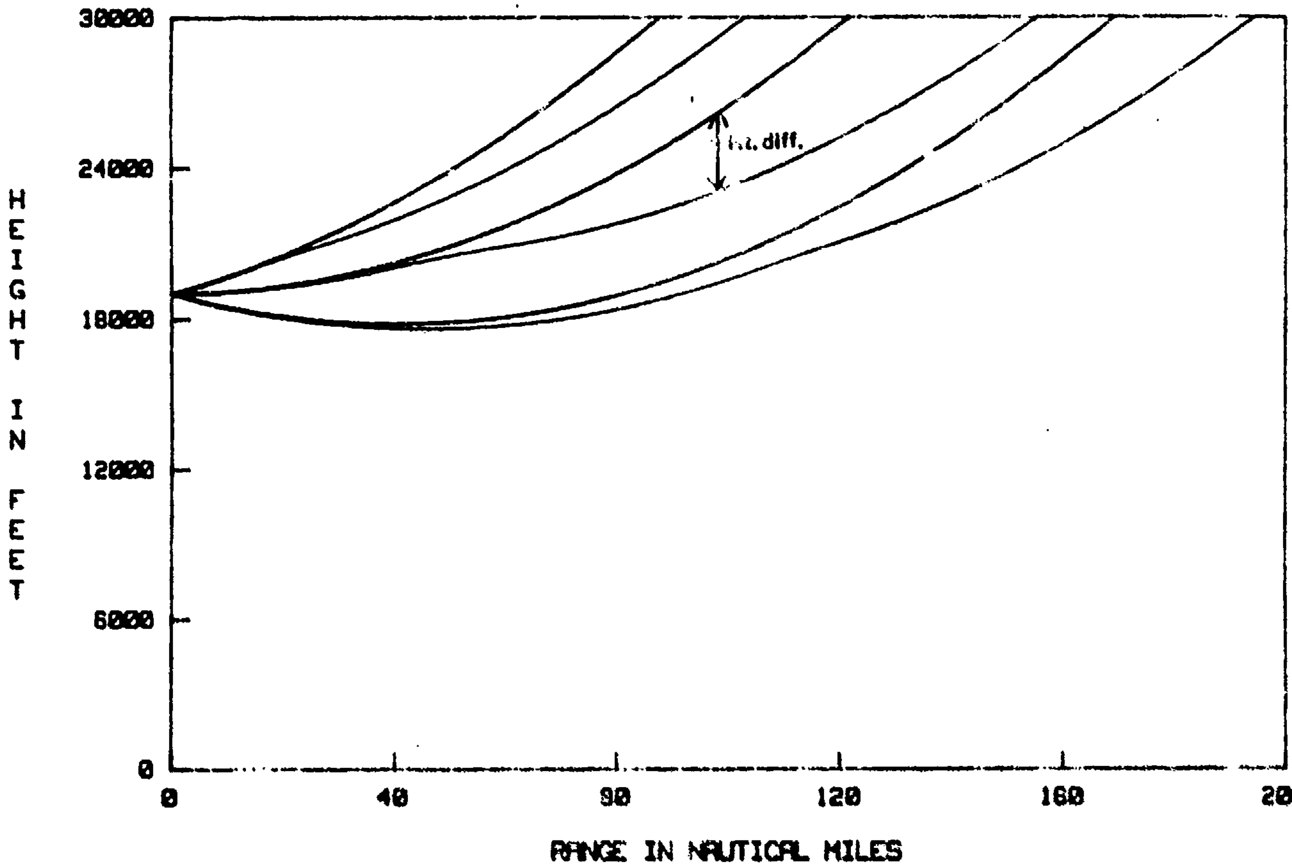


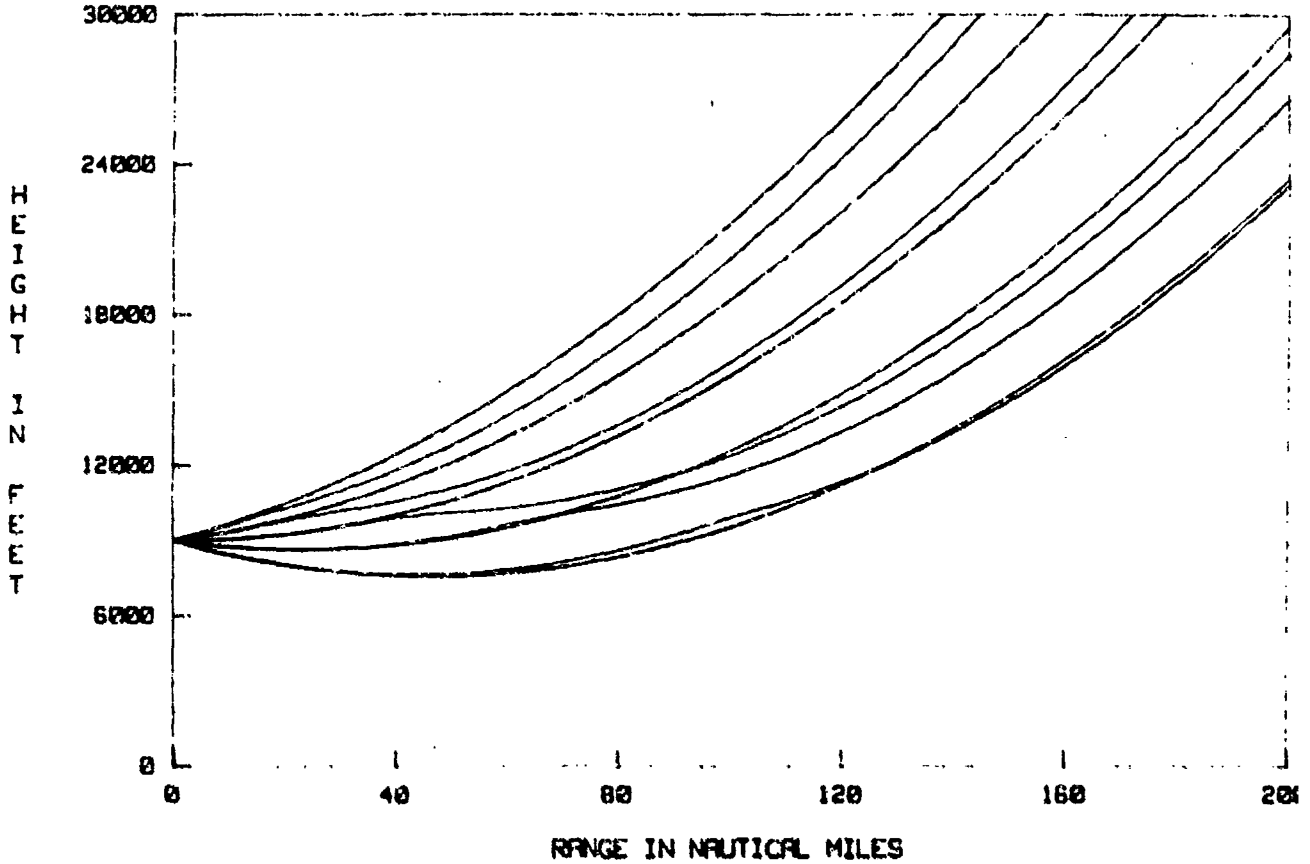
Figure 6. Error display for elevated duct – 200 rays.



HEIGHT(ft)	M-UNITS
.0	350.0
20000.0	1070.0
21000.0	1050.0
22000.0	1086.0

ANTENNA HEIGHT (ft): 19000.0
LOWER ELEVATION ANGLE (in mrad): -10.0
UPPER ELEVATION ANGLE (in mrad): 10.0
NUMBER OF RAYS: 3

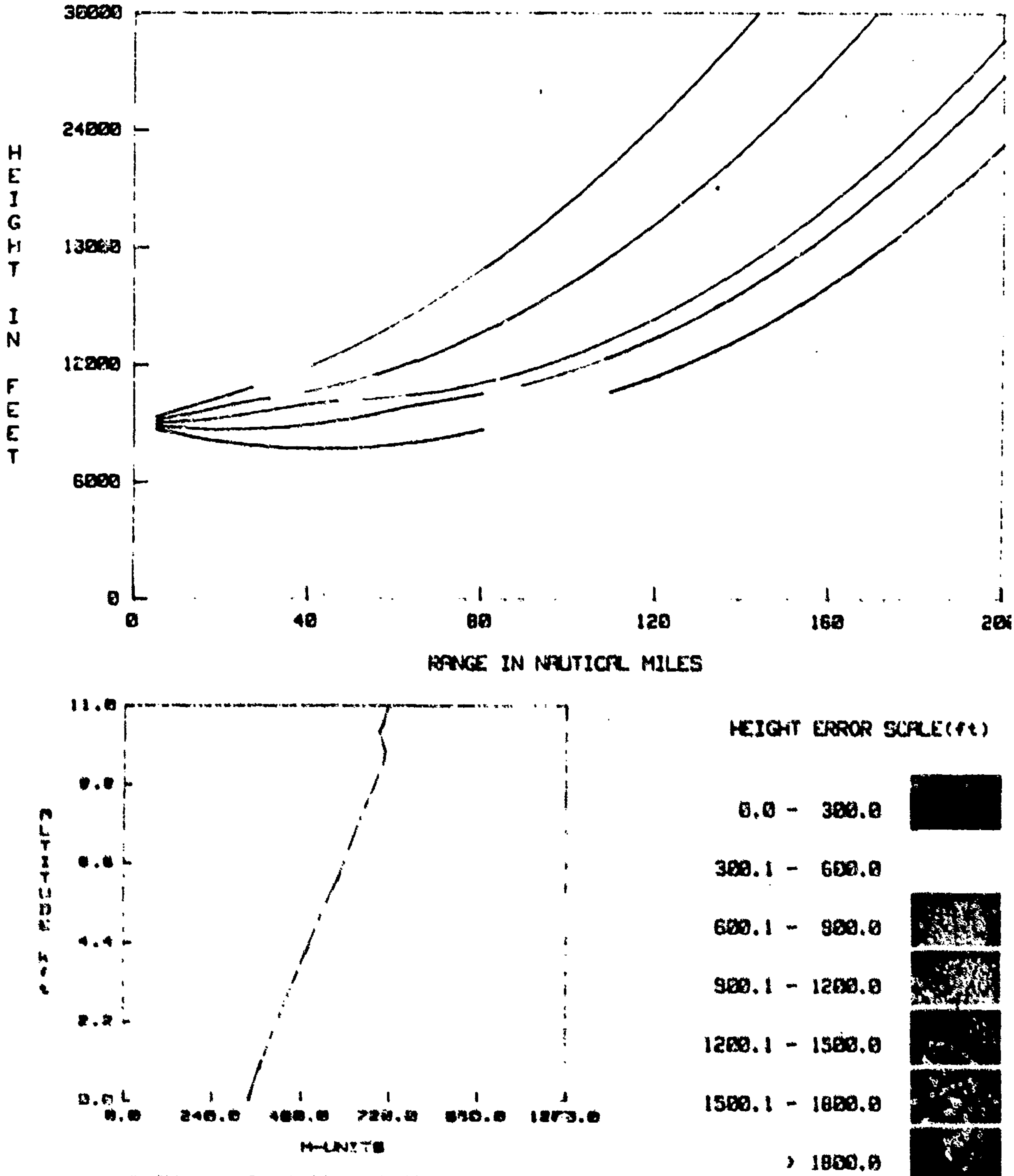
Figure 7. Ray-trace display for elevated duct of figure 6 (red) and standard atmosphere (black).



HEIGHT(ft)	M-UNITS
16.0	339.8
669.0	365.6
4125.0	492.3
6105.0	575.2
9418.0	702.7
9726.0	707.5
10256.0	688.7
10449.0	698.0
11008.0	715.7
12759.0	792.5
14409.0	861.5
20207.0	1109.9

ANTENNA HEIGHT (ft): 9000.0
LOWER ELEVATION ANGLE (in mrad): -10.0
UPPER ELEVATION ANGLE (in mrad): 10.0
NUMBER OF RAYS: 5

Figure 8. Ray-trace display for elevated duct (red) and standard atmosphere (black).



ANTENNA HEIGHT (ft): 9000.0
LOWER ELEVATION ANGLE (in mrad): -10.0
UPPER ELEVATION ANGLE (in mrad): 10.0
NUMBER OF RAYS: 5

Figure 9. Error display for elevated duct of figure 8 — five rays.

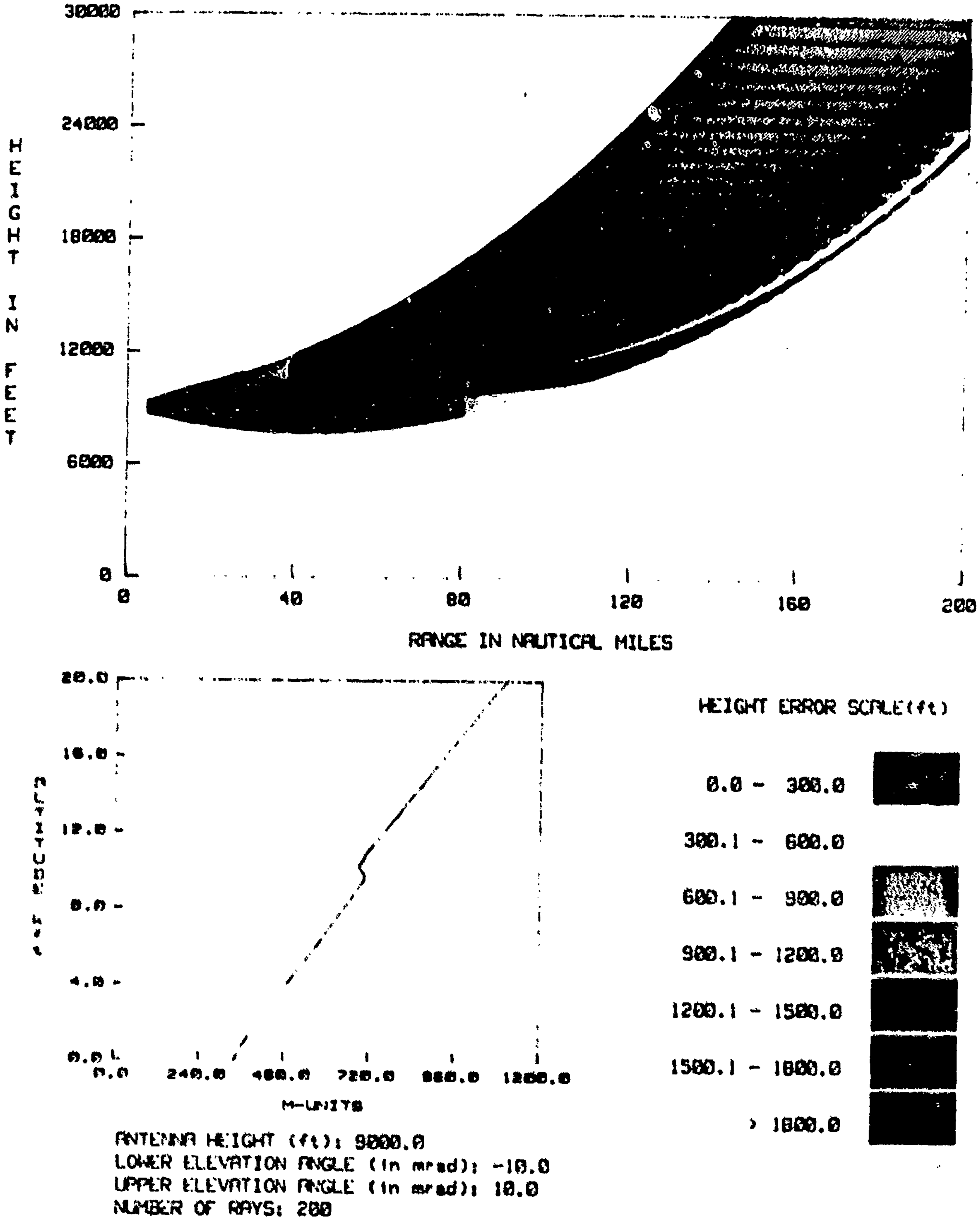


Figure 10. Error display for elevated duct of figure 8 — 200 rays.

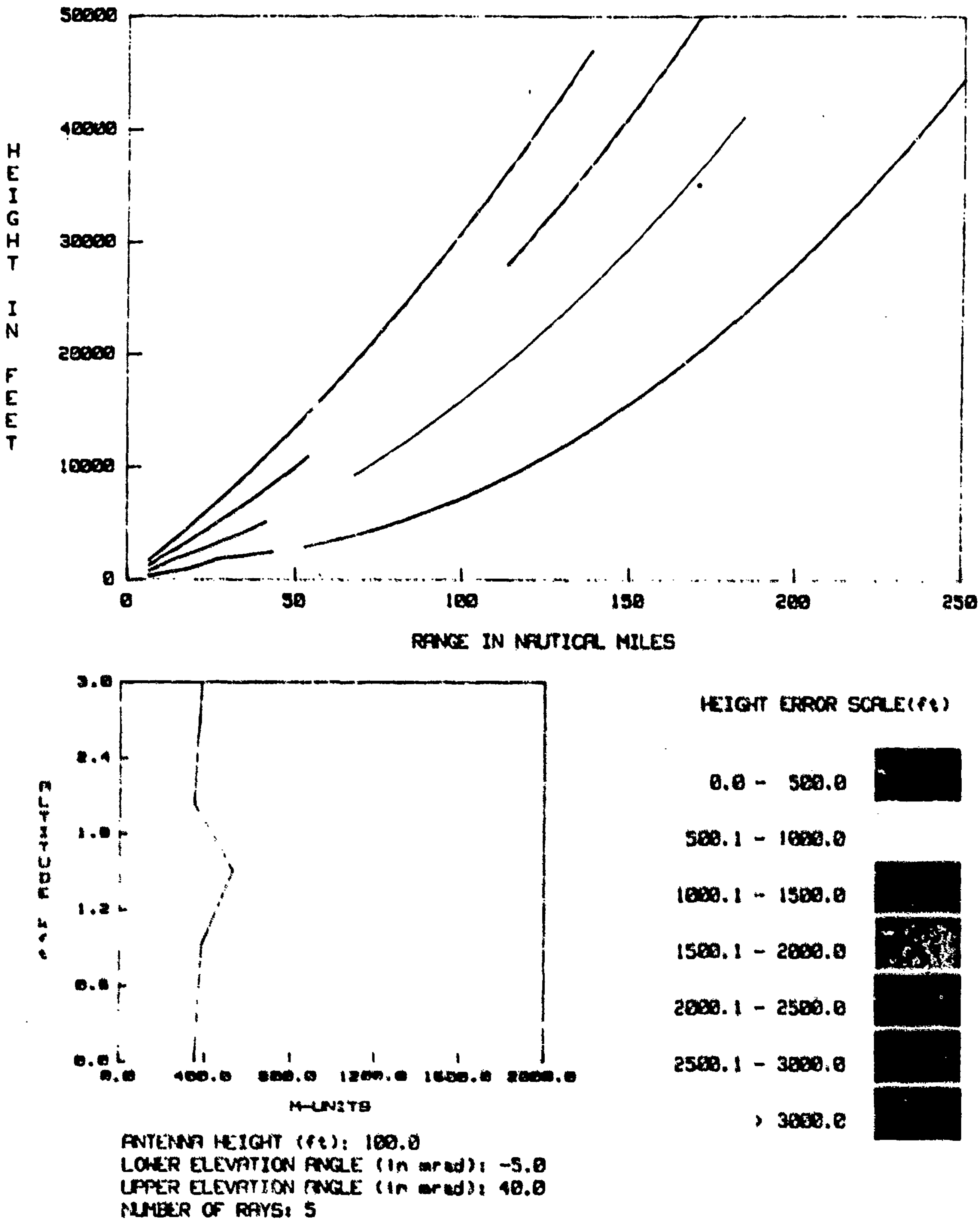
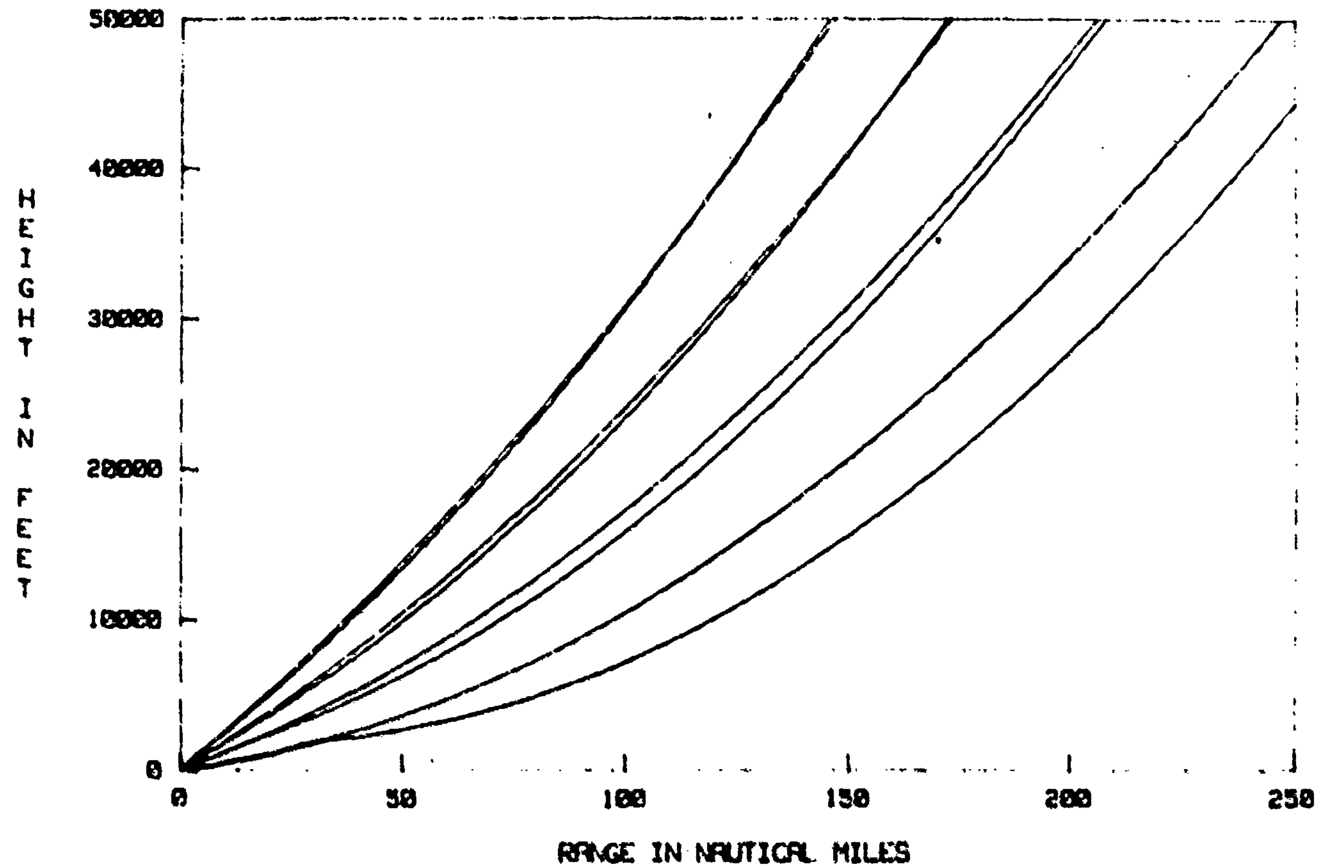


Figure 11. Error display for surface-based duct — five rays.



HEIGHT(ft)	M-UNITS	HEIGHT(ft)	M-UNITS
13.0	355.6	15639.0	913.4
931.0	385.6	19748.0	1008.9
1506.0	531.1	32300.0	1642.5
2034.0	349.3	38209.0	1906.3
2681.0	372.3		
3575.0	406.1		
4417.0	440.3		
4857.0	475.3		
5606.0	492.1		
6514.0	525.2		
6935.0	543.9		
7412.0	563.6		
8536.0	610.5		
9625.0	659.8		
13968.0	852.3		

ANTENNA HEIGHT (ft): 100.0
LOWER ELEVATION ANGLE (in mrad): -5.0
UPPER ELEVATION ANGLE (in mrad): 40.0
NUMBER OF RAYS: 5

Figure 12. Ray-trace display for surface-based duct of figure 11 (red) and standard atmosphere (black).

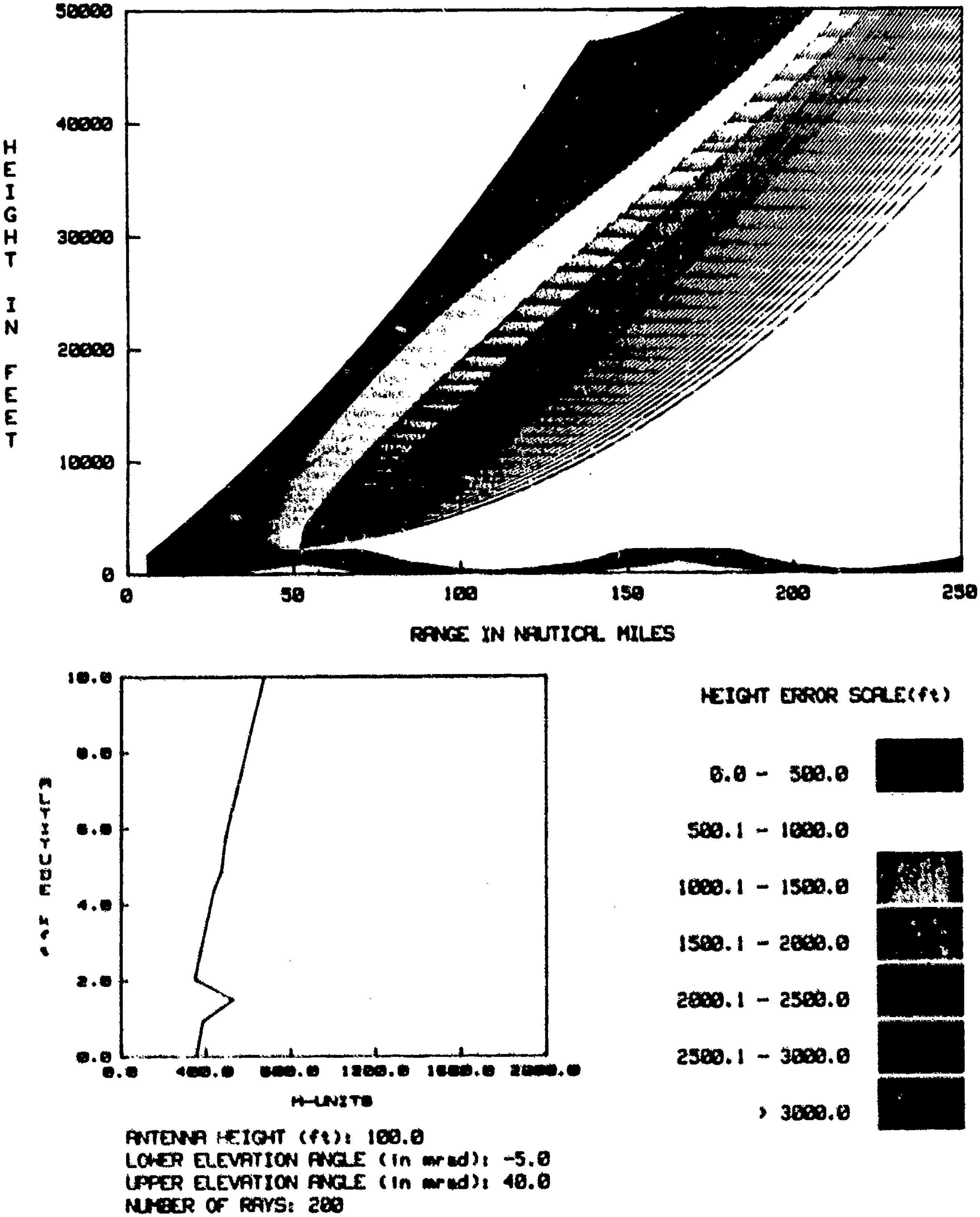


Figure 13. Error display for surface-based duct of figure 11 -- 200 rays.

CONCLUSIONS

A height-finder radar altitude-error display has been developed on the HP9000 – 500-series computer based on traditional ray-tracing concepts. This display shows altitude error compared to a standard atmosphere using a color scale superimposed on a traditional altitude-versus-range ray trace for arbitrary, piecewise, linear refractivity-versus-altitude profiles. Since this effort only considered alternate displays of well-established ray-tracing theory, no attempt was made to validate the accuracy of any of the resulting displays.

RECOMMENDATION

The altitude-error displays reported here should be incorporated into the coverage diagram of the Integrated Refractive Effects Prediction System (IREPS).

APPENDIX A: COMPUTER PROGRAM LISTING

```
c      ************** Variables used in program **********************
c
c      dmdh:         35 element array containing the gradients of the
c                    profile.
c      erropt:       Character string indicating if absolute ('a' or 'A')
c                    or percent ('p' or 'P') for color scale was chosen.
c      aber:         Absolute error increment in meters.
c      per:          Percent error increment.
c      fildes:       File descriptor for starbase graphics routine.
c      index:        Integer indicating color to be used for plotting
c                    (1-7).
c      mvfl:         Logical flag indicating when to move to the next
c                    height-range point for plotting onto hardcopy when
c                    changing pen colors.
c      ht:           Antenna height in meters.
c      hmax:         Maximum height for plot display in meters.
c      rmax:         Maximum range for plot display in kilometers.
c      orh:          33 element array containing heights of the original
c                    profile.
c      orm:          33 element array containing m-units of the original
c                    profile.
c      h:            35 element array containing heights of the new
c                    profile.
c      im:           Counter indicating what color is currently being used
c                    for hardcopy plotting.
c      plot:         Logical flag indicating if plotting onto hardcopy,
c                    (T-plot,F-don't plot).
c      name:         Character string containing the data filename to be
c                    read.
c      nlvl:         Number of new levels after extrapolation to the
c                    surface and maximum height.
c      ilvl:         Number of levels in the original profile.
c      m:            33 element array containing m-units of the new
c                    profile.
c      f7:           Character string indicating if 'AUTODUMP' is set to on
c                    or off.
c      angle:        Array containg initial elevation angles (300 maximum).
c      hun:          Character string indicating what height units the
c                    profile was stored in.
c      reun:         Character string indicating if refractivity values
c                    were stored in m-units or n-units.
c      frstime:      Logical flag indicating first height-range point
c                    calculated for new ray.
c      sor:          Starting range for hardcopy plotting.
c      soh:          Starting height for hardcopy plotting.
c      frst:         Logical flag indicating first storage of height-range
c                    point for interpolation between color transitions.
c      ipro_print:   Integer indicating type of profile printout for
c                    hardcopy; 1-graph, 2-number.
c      twice:        Logical flag indicating if program has been run at
c                    least once.
c
        common/errvar/dmdh(35),erropt,aber,per,fildes,index,mvfl,
       +ht,hmax,rmax,rinc,orh(33),orm(33),h(35),im,plot,name,
       +nlvl,ilvl,m,f7,angle(300),hun,reun,frstime,sor,soh,frst,
       +ipro_print,twice
```

```
      integer*4 fildes,index
      real m(35)
      character*3 f7
      character*1 erropt,hun,reun
      character*14 name
      logical mvfl,plot,frstime,frst,logical,twice
```

```
c      ************************ PROGRAM HFERR ****************************
c
c      Purpose:  This is the main program that will begin all procedures
c                for ray tracing.
c
c      Glossary:
c           opst      Character string indicating if the 'option' or
c                     'backup' key was hit.
c           outst     Character string variable.
c           optn      Real value of option number chosen by user.
c           ioptn     Integer value of option number chosen by user.
c           flag      Logical flag indicating if the user has gone
c                     through the edit routine.
c           profile   Array containing all data filenames.
c           ifil      Number of data filenames in profile.
c           rpick     Real value of number of environmental data file
c                     chosen.
c           ipick     Integer value of number of environmental data file
c                     chosen.
c           rlowea    Lower elevation angle in mrad.
c           upea      Upper elevation angle in mrad.
c           rplot     Number of rays to be plotted.
c           irplot    Integer value of number of rays to be plotted.
c
c      *******************************************************************
c
       program hferr
       include '/usr/include/starbase.f1.h'
       include '/usr/include/starbase.f2.h'
$INCLUDE 'errvar'
       character*3 fun
       character*8 opst,kgt
       character*14 profile(50)
       character*16 dmpfrm
       character*80 dummy,outst
       integer*4 status
       logical flag,duct,us_flag
       ir=5
       iw=6
       call kyinit(ir,iw,15)
       twice=.false.
       us_flag=.false.
       ipro_print=0
c
c   ********** Clear alphanumerics screen and write options list. ******
c
 10    call kycrsor(0,0,-1)
       write(iw,'("1 -- Delete an environmental data file")')
       write(iw,'("2 -- Add an environmental data file")')
       write(iw,'("3 -- Edit an environmental data file")')
       write(iw,'("4 -- Altitude Error Display")')
       if(twice) write(iw,'("5 -- Display for same profile")')
       write(iw,*)
       call kyenter("option number (or 'end') ",0,1.,5.,-1.,optn,opst)
       if((opst.eq.'backup').or.(opst.eq.'option')) goto 10
       write(iw,*)
```

```
      ioptn=nint(optn)
      if(ioptn.eq.4) twice=.false.
      if(option.eq.-1)opst='end'
      if((opst.eq.'end').or.(opst.eq.'END')) then
         goto 20
      else
         if(.not.twice) call renew
      end if
 30   flag=.false.
      if((ioptn.eq.1).or.(ioptn.eq.2).or.(ioptn.eq.3)) then
         call edit(ioptn,opst,flag,iw)
         if((opst.eq.'option').or.(opst.eq.'backup')) goto 10
      end if
      if(flag) goto 10
      if(ioptn.eq.5) then
         us_flag=.true.
         call putin(rl,ru,irplot,filvar,fun)
         goto 15
      end if
c
c   ************* List existing environmental data files. *************
c
      call lsfiles('.prof',profile,ifil)
      if(ifil.eq.0) then
         write(iw,'(a1)')char(7)
         call kystent("** No data files exist.  Press 'RETURN' to create
     + new file. **",-1,'*',' ',outst,opst)
         if((opst.eq.'backup').or.(opst.eq.'option')) goto 10
         ioptn=2
         goto 30
      end if
      write(iw,*)
c
c   ************ Begin  writing prompt strings for user input. ********
c
 40   call kyenter("number of environmental data file ",0,1.,real(ifil),
     +0.,rpick,opst)
      if((opst.eq.'backup').or.(opst.eq.'option')) goto 10
      ipick=nint(rpick)
      name=profile(ipick)
      write(iw,*)
 50   call kyenter("maximum height for display",1,1.,50000.,0.,hmax,
     +opst)
      if(opst.eq.'backup') goto 40
      if(opst.eq.'option') goto 10
      write(iw,*)
 60   call kyenter("maximum range for display",2,20.,1000.,0.,rmax,
     +opst)
      if(opst.eq.'backup') goto 50
      if(opst.eq.'option') goto 10
      write(iw,*)
 70   call kyenter("antenna height",1,1.,50000.,0.,ht,opst)
      if(opst.eq.'backup') goto 60
      if(opst.eq.'option') goto 10
      write(iw,*)
 80   call kyenter("lower elevation angle in mrad",0,-100.,100.,101.,rl,
```

```
     +opst)
      if(opst.eq.'backup') goto 70
      if(opst.eq.'option') goto 10
      write(iw,*)
 90   call kyenter("upper elevation angle in mrad",0,-100.,100.,101.,ru,
     +opst)
      if(opst.eq.'backup') goto 80
      if(opst.eq.'option') goto 10
      write(iw,*)
 100  call kyenter("number of rays to be plotted",0,1.,300.,0.,rplot,
     +               opst)
      if(opst.eq.'backup') goto 90
      if(opst.eq.'option') goto 10
      irplot=nint(rplot)
      write(iw,*)
 110  call kystent("A or P for absolute or percent error",0,'A,a,P,p',
     +'a',erropt,opst)
      if(opst.eq.'backup') goto 100
      if(opst.eq.'option') goto 10
      write(iw,*)
      if((erropt.eq.'a').or.(erropt.eq.'A')) then
 120     call kyenter("number for error increment",1,1.,5000.,0.,aber,
     +opst)
         if(opst.eq.'backup') goto 110
         if(opst.eq.'option') goto 10
         write(iw,*)
      else
 130     call kyenter("percent error increment",0,1.,20.,5.,per,opst)
         if(opst.eq.'backup') goto 110
         if(opst.eq.'option') goto 10
         write(iw,*)
      end if
      call readda(ipick,profile,hun,reun)
 15   call kyread(7,f7)
c
c    ********** Call INITIAL to set up arrays and variables used in
c    the ray tracing subroutines. **********************************
c
      call initial(rl,ru,irplot,duct,forbid)
c
c    ********** Call GRAPH to initialize graphics screen and draw axes
c    for plotting. *************************************************
c
      call kyread(6,kgt)
      if((ipro_print.eq.2).or.(ipro_print.eq.1)) kgt='on'
      if(kgt.eq.'off') then
         plot=.false.
      else if(kgt.eq.'on') then
         plot=.true.
         if((ipro_print.ne.1).and.(ipro_print.ne.2)) then
            call kyenter("profile printout:  1 - graph ; 2 - numbers",
     +         0,1.,2.,1.,pro_print,opst)
            ipro_print=nint(pro_print)
         end if
      end if
 25   call graph(rl,ru,irplot)
```

```
c
c    ******************** Start of main calculations. ***************
c
       call actapp(irplot,duct,forbid)
       write(iw,*)
c
c    *********************** Begin dump. ****************************
c
       status=gclose(fildes)
       twice=.true.
       if(plot) goto 10
       call kystent("Press 'AUTODUMP-ON' and 'RETURN' for hardcopy",
      +-1,'*',' ',outst,opst)
       if(opst.eq.'option') goto 10
       call kyread(6,kgt)
       if(kgt.eq.'on') then
          call kyenter("profile printout:  1 - graph ; 2 - numbers",
      +   0,1.,2.,1.,pro_print,opst)
          if(opst.eq.'option') goto 10
          ipro_print=nint(pro_print)
          plot=.true.
          goto 25
       else
          goto 10
       end if
 20    continue
       if(us_flag) call system('rm usin')
       call kyterm
       end
```

```
c   ************************** SUBROUTINE ACTAPP **************************
c
c      Purpose:  This begins the main loop of the program.  It checks for
c                the transmitter height level and begins the ray-tracing
c                subroutines.
c
c      Glossary:
c
c          duct     Logical flag indicating if the transmitter height is
c                   at a break point.
c          irays    Number of rays to be used in plotting.
c
c    *******************************************************************
c
      subroutine actapp(irays,duct,forbid)
$INCLUDE 'errvar'
      logical duct
c
c   ******* Check what level the transmitter height is at. *************
c
      do ix=1,nlvl
         if((h(ix).le.ht).and.(ht.le.h(ix+1))) goto 40
      end do
 40   continue
c
c   **************** Begin main loop. ***********************************
c
      im=1
      if(plot) then
         call seven(irays,duct,forbid,ix)
      else
         call regular(irays,duct,forbid,ix)
      end if
      return
      end
c
c      ********************** SUBROUTINE STAND ***************************
c
c      Purpose:  This routine performs a ray trace for a standard atmos-
c                phere.  It begins at the transmitter height and traces
c                to the specified range given by the current non-standard
c                atmosphere being used with the same initial elevation
c                angle.
c
c      Glossary:
c          hs    Height traced to at range rp.
c          hdif  Difference in height between the ray traced in a stan-
c                dard atmosphere and that traced in a non-standard atmos-
c                phere at the same range.
c
c    *******************************************************************
c
      subroutine stand(fixang,rp,hp)
$INCLUDE 'errvar'
      rf=rp
      rbef=0.
```

```
        hbef=ht
        alpha=fixang
c
c    ************ Begin trace for down-going rays. ********************
c
        if(alpha.lt.0.) then
           alphap=alpha+1.18e-4*rf
           if(alphap.ge.0.) then
              alphap=0.
              range=rbef-alpha/1.18e-4
              hs=hbef-alpha**2/2.36e-7
              alpha=alphap
              hbef=hs
              rbef=range
              goto 10
           end if
           hs=hbef+(alphap**2-alpha**2)/2.36e-7
c
c    ******* If the height is negative, then ray is reflected. *********
c
           if(hs.lt.0.) then
              alphap=-SQRT(alpha**2+2.36e-7*(-hbef))
              range=rbef+(alphap-alpha)/1.18e-4
              alphap=-alphap
              alpha=alphap
              rbef=range
              hbef=0.
              goto 10
           end if
           goto 20
        end if
c
c    ************  Begin trace for up-going rays. ********************
c
 10     if(alpha.ge.0.) then
           alphap=alpha+1.18e-4*(rf-rbef)
           hs=hbef+(alphap**2-alpha**2)/2.36e-7
        end if
 20     continue
        hdif=ABS(hp-hs)
c
c    ************ Calling routine to determine color according to error
c    increment scale. ***********************************************
c
        rng=0.
        hyt=0.
c
        call interpol(hdif,rp,hp,rng,hyt,hs)
        frst=.false.
        if(plot) then
           call stepfixplt(hdif,hs,rp,hp,rng,hyt)
        else
           call stepfixscr(hdif,hs,rp,hp,rng,hyt)
        end if
        return
        end
```

```
c   ********************** SUBROUTINE DOWN ******************************
c
c      Purpose:  This routine performs a ray trace for down-going rays.
c
c      Glossary:
c
c        rbef      Range before calculation.
c        rp        Range after calculation.
c        rinc      Range increment.
c        hbef      Height before calculation.
c        hp        Height after calculation.
c        ij        Height level counter.
c        alpha     Angle before calculation.
c        alphap    Angle after calculation.
c
c   *********************************************************************
c
$OPTION ONEYRIP
      subroutine down(alpha,hbef,ij,rbef,rp,hp,fixang)
      include '/usr/include/starbase.f1.h'
      include '/usr/include/starbase.f2.h'
$INCLUDE 'errvar'
c
c   *************** Begin loop to trace from transmitter height to the
c   first level or until a minimum has been reached. ********************
c
      do while((ij.ge.1).and.(alpha.le.0.))
         rp=rbef+rinc
         if(rp.ge.rmax) goto 10
         alphap=alpha+dmdh(ij)*(rp-rbef)
c
c   ************ If the new angle calculated is positive then a minimum
c   has been reached.  ALPHAP is set to 0. *****************************
c
         if(alphap.ge.0.) then
            alphap=0.
            rp=rbef-alpha/dmdh(ij)
            hp=hbef-alpha**2/2.e-3/dmdh(ij)
         end if
         hp=hbef+(alphap**2-alpha**2)/2.e-3/dmdh(ij)
         if(hp.lt.h(ij)) then
            hp=h(ij)
            rad=alpha**2+2.e-3*dmdh(ij)*(hp-hbef)
c
c   ************ If RAD is negative then a minimum has been reached.
c   ALPHAP is set to 0. ********************************************
c
            if(rad.le.0.) then
               alphap=0.
               rp=rbef-alpha/dmdh(ij)
               hp=hbef-alpha**2/2.e-3/dmdh(ij)
            else
               alphap=-SQRT(rad)
               rp=rbef+(alphap-alpha)/dmdh(ij)
            end if
            ij=ij-1
```

```
              end if
      c
      c  ********* Once a specific height and range have been calculated
      c  subroutine STAND is called to calculate the height at the same range
      c  RP for a standard atmosphere. ***************************************
      c
              if(rp.ge.rmax) goto 10
              call stand(fixang,rp,hp)
              rbef=rp
              hbef=hp
              alpha=alphap
              if(alpha.eq.0.) goto 10
           end do
           if(ij.eq.0.) goto 30
      c
      c   ************ Trace to exactly the maximum range. ******************
      c
   10      if(rp.ge.rmax) then
              rp=rmax
              alphap=alpha+dmdh(ij)*(rp-rbef)
              hp=hbef+(alphap**2-alpha**2)/2.e-3/dmdh(ij)
              call stand(fixang,rp,hp)
           end if
   30      return
           end
```

```
c      ******************** SUBROUTINE EDIT ******************************
c
c      This subroutine performs all the editing operations of the program.
c
       subroutine edit(ioptn,opst,flag,iw)
       character*8 opst,outst
       character*14 filena(50),dumname
       character*80 dum
       logical flag
       flag=.true.
c
c      ************** Specified files are deleted or put into the vi
c      editor for editing. ****************************************
c
       if((ioptn.eq.1).or.(ioptn.eq.3)) then
          call lsfiles('.prof',filena,ifil)
          if(ifil.eq.0) then
             write(iw,'(a1)')char(7)
             call kystent("*** No data files exist.  Press 'RETURN' to cr
     +eate new file. ***",-1,'*',' ',outst,opst)
             if((opst.eq.'backup').or.(opst.eq.'option')) goto 10
             ioptn=2
             goto 20
          end if
          write(iw,*)
          call kyenter("file number",0,1.,real(ifil),0.,rnum,opst)
          if((opst.eq.'option').or.(opst.eq.'backup')) goto 10
          inum=nint(rnum)
       end if
 20    if(ifil.eq.0) ifil=1
c
c      ********* For adding an environmental file the user is put into
c      the vi editor. ********************************************
c
       if(ioptn.eq.2) then
          call system('cp .dprof dprof')
          call system('vi dprof')
          open(3,FILE='dprof')
          read(3,'(3(a80/),a80)') (dum,i=1,4)
          read(3,'(a14)') filena(ifil)
          close(3)
          call system('mv dprof .prof/'//filena(ifil)//char(0))
       else if(ioptn.eq.1) then
          call system('rm .prof/'//filena(inum)//char(0))
       else
          call system('vi .prof/'//filena(inum)//char(0))
          open(3,FILE='.prof/'//filena(inum))
          read(3,'(3(a80/),a80)')(dum,i=1,4)
          read(3,'(a14)')dumname
          if(dumname.ne.filena(inum)) call system('mv .prof/'//filena
     +    (inum)//' .prof/'//dumname//char(0))
          close(3)
       end if
 10    return
       end
```

```
c     ************************ SUBROUTINE INITIAL ********************
c
c     Purpose:  This subroutine calculates all the necessary constants
c               and arrays for use in actual ray trace calculations.
c
c     Glossary:
c          rplot      Real value of the number of rays to be plotted.
c          irplot     Integer value of the number of rays to be plotted.
c          forbid     Angular region of unallowable ray traces.
c          duct       Flag indicating if the transmitter height is at
c                     a breakpoint.
c          check      Flag indicating if hmax is equal to any of the
c                     heights in the profile.
c          rl         Lower elevation angle in mrad.
c          rfirst     Lower elevation angle in rad.
c          ru         Upper elevation angle in mrad.
c          rlast      Upper elevation angle in rad.
c          ainc       Incremental angle value for alpha array.
c          alpha      Array containing all initial launch angles.
c          rinc       Range increment.
c          rmetperdot Meters per dot on screen.
c
c     ****************************************************************
c
      subroutine initial(rl,ru,irplot,duct,forbid)
$INCLUDE 'errvar'
      logical duct,check
c
c  ************* Convert lower and upper angles to radians and store in
c  an array in incremental values corresponding to the number of rays t
c  by plotted. ********************************************************
c
      rfirst=rl*1.e-3
      rlast=ru*1.e-3
      ainc=(rlast-rfirst)/(irplot-1 )
      angle(1)=rfirst
      do i=2,irplot
         angle(i)=angle(i-1)+ainc
      end do
      angle(irplot)=rlast
      rinc=rmax/50.
      sor=.025*rmax
c
c  ************* Convert height and index array to meters and m-units
c  if necessary. *****************************************************
c
      do i=1,ilvl
         if(hun.eq.'F') then
            h(i)=orh(i)/3.280839
         else
            h(i)=orh(i)
         end if
         if(reun.eq.'N') then
            m(i)=orm(i)/6.371
         else
            m(i)=orm(i)
```

```
            end if
         end do
         nlvl=ilvl
c
c  ************* If the first height level is not 0. then an extrapola-
c  tion is done to find the m-unit value at the surface. *************
c
         if(h(1).ne.0.) then
            nlvl=nlvl+1
            surf=h(1)/(h(2)-h(1))*(m(1)-m(2))
            m(nlvl)=m(1)+surf
         end if
c
c  ********** Each height level is checked to see if it matches the
c  maximum height. If not, then the maximum height is included in the
c  array and the m-unit value at that height is found. **************
c
         check=.false.
         do i=1,ilvl
            if((h(i).le.hmax+1.e-5).and.(h(i).ge.hmax-1.e-5)) check=.true.
         end do
         if(.not.check) then
            nlvl=nlvl+1
            h(nlvl)=hmax
         end if
c
c  **************** The array is sorted. ****************************
c
         do 10 i=1,nlvl-1
            do 20 j=i+1,nlvl
               if(h(j).gt.h(i)) goto 20
               dum=h(i)
               h(i)=h(j)
               h(j)=dum
               dumm=m(i)
               m(i)=m(j)
               m(j)=dumm
   20       end do
   10    end do
c
c  *********** If HMAX is between two height levels then the m-unit
c  value is found at HMAX. ****************************************
c
         if(hmax.ne.h(nlvl)) then
            do j=1,nlvl
               if(h(j).eq.hmax) then
                  bet=(h(j)-h(j-1))/(h(j+1)-h(j-1))*(m(j+1)-m(j-1))
                  m(j)=m(j-1)+bet
               end if
            end do
c
c   ******* The gradient at each level is calculated and stored. ******
c
         else
            rint=(h(nlvl)-h(nlvl-1))/(h(nlvl-1)-h(nlvl-2))*(m(nlvl-1)-
     *      m(nlvl-2))
```

```
            m(nlvl)=m(nlvl-1)+rint
         end if
         do i=1,nlvl-1
            dmdh(i)=(m(i+1)-m(i))/(h(i+1)-h(i))*1.e-3
            if(ABS(dmdh(i)).lt.1e-6) dmdh(i)=1.e-6
         end do
         rmetperdot=rmax/390.
c
c     ************** The transmitter height is checked to see if it is
c     at a breakpoint.  If so, the unallowable angular range is
c     calculated. ******************************************************
c
         forbid=0.
         duct=.false.
         do i=1,nlvl
            if((h(i).le.ht+1.e-6).and.(h(i).ge.ht-1.e-6)) then
               if((dmdh(i-1).gt.0.).and.(dmdh(i).lt.0.)) forbid=
      +        SQRT(-2.e-3*dmdh(i)*rmetperdot)
            end if
         end do
         if(forbid.ne.0.) duct=.true.
         return
         end
```

```
c
c     ******************** SUBROUTINE INTERPOL *************************
c
c     Purpose:  This subroutine checks if a transition from one color
c               to the next has been reached.
c
c     ******************************************************************
c
      subroutine interpol(dif_2,rng_2,hp_2,rng,hyt,hs_2)
$INCLUDE 'errvar'
      save dif_1,rng_1,hp_1,hs_1
      if(frst) then
         dif_1=dif_2
         rng_1=rng_2
         hp_1=hp_2
         hs_1=hs_2
         goto 10
      end if
      if((erropt.eq.'a').or.(erropt.eq.'A')) then
         if(((dif_1.le.aber).and.(dif_2.gt.aber)).or.((dif_2.le.aber)
     +   .and.(dif_1.gt.aber))) then
            call int(dif_1,dif_2,rng_1,rng_2,hp_1,hp_2,rng,hyt,aber)
         else if(((dif_1.le.2.*aber).and.(dif_2.gt.2.*aber)).or.
     +   ((dif_2.le.2.*aber).and.(dif_1.gt.2.*aber))) then
            call int(dif_1,dif_2,rng_1,rng_2,hp_1,hp_2,rng,hyt,2.*aber)
         else if(((dif_1.le.3.*aber).and.(dif_2.gt.3.*aber)).or.
     +   ((dif_2.le.3.*aber).and.(dif_1.gt.3.*aber))) then
            call int(dif_1,dif_2,rng_1,rng_2,hp_1,hp_2,rng,hyt,3.*aber)
         else if(((dif_1.le.4.*aber).and.(dif_2.gt.4.*aber)).or.
     +   ((dif_2.le.4.*aber).and.(dif_1.gt.4.*aber))) then
            call int(dif_1,dif_2,rng_1,rng_2,hp_1,hp_2,rng,hyt,4.*aber)
         else if(((dif_1.le.5.*aber).and.(dif_2.gt.5.*aber)).or.
     +   ((dif_2.le.5.*aber).and.(dif_1.gt.5.*aber))) then
            call int(dif_1,dif_2,rng_1,rng_2,hp_1,hp_2,rng,hyt,5.*aber)
         else if(((dif_1.le.6.*aber).and.(dif_2.gt.6.*aber)).or.
     +   ((dif_2.le.6.*aber).and.(dif_1.gt.6.*aber))) then
            call int(dif_1,dif_2,rng_1,rng_2,hp_1,hp_2,rng,hyt,6.*aber)
         end if
       else if((erropt.eq.'p').or.(erropt.eq.'P')) then
         if((hs_1.eq.0.).or.(hs_2.eq.0.)) then
            hs_1=1.
            hs_2=1.
         end if
         perc_1=(dif_1/hs_1)*100.
         perc_2=(dif_2/hs_2)*100.
         if(((perc_1.le.per).and.(perc_2.gt.per)).or.((perc_2.le.per)
     +   .and.(perc_1.gt.per))) then
            call int(perc_1,perc_2,rng_1,rng_2,hp_1,hp_2,rng,hyt,per)
         else if(((perc_1.le.2.*per).and.(perc_2.gt.2.*per)).or.
     +   ((perc_2.le.2.*per).and.(perc_1.gt.2.*per))) then
            call int(perc_1,perc_2,rng_1,rng_2,hp_1,hp_2,rng,hyt,2.*per)
         else if(((perc_1.le.3.*per).and.(perc_2.gt.3.*per)).or.
     +   ((perc_2.le.3.*per).and.(perc_1.gt.3.*per))) then
            call int(perc_1,perc_2,rng_1,rng_2,hp_1,hp_2,rng,hyt,3.*per)
         else if(((perc_1.le.4.*per).and.(perc_2.gt.4.*per)).or.
     +   ((perc_2.le.4.*per).and.(perc_1.gt.4.*per))) then
```

```
                call int(perc_1,perc_2,rng_1,rng_2,hp_1,hp_2,rng,hyt,4.*per)
             else if(((perc_1.le.5.*per).and.(perc_2.gt.5.*per)).or.
     +       ((perc_2.le.5.*per).and.(perc_1.gt.5.*per))) then
                call int(perc_1,perc_2,rng_1,rng_2,hp_1,hp_2,rng,hyt,5.*per)
             else if(((perc_1.le.6.*per).and.(perc_2.gt.6.*per)).or.
     +       ((perc_2.le.6.*per).and.(perc_1.gt.6.*per))) then
                call int(perc_1,perc_2,rng_1,rng_2,hp_1,hp_2,rng,hyt,6.*per)
             end if
          end if
          dif_1=dif_2
          rng_1=rng_2
          hp_1=hp_2
          hs_1=hs_2
 10       return
          end
c
c         ******************* SUBROUTINE INT ******************************
c
c         Purpose:  This subroutine interpolates the transition from one
c                   height error increment (color) to the next.
c
c         *****************************************************************
c
          subroutine int(dum_1,dum_2,rng_1,rng_2,hp_1,hp_2,rng,hyt,fxdif)
          rin=(fxdif-dum_1)/(dum_2-dum_1)*(rng_2-rng_1)
          rng=rng_1+rin
          hin=(rng-rng_1)/(rng_2-rng_1)*(hp_2-hp_1)
          hyt=hp_1+hin
          return
          end
```

```
c     ********* Subroutine LSFILES produces a two-column list, preceded
c     by a number, of all data files stored in directory 'DIR'.
c     The maximum number of data files 46. ***************************
c
c     *****************************************************************
c
c     Glossary:
c         ifil    Counter for number of files in directory.  Will be
c                 zero if no files in that directory.
c         ire     One-half of ifil to print out on screen a two-column
c                 list of filenames
c         name    46 element array - stores filenames
c
c     *****************************************************************
c
      subroutine lsfiles(dir,name,ifil)
      character*14 name(46)
      character*(*) dir
      call system('ls '//dir//')@datafiles@'//char(0))
      open(1,FILE='@datafiles@')
      ifil=0
      do j=1,46
         read(1,'(a14)',END=10)name(j)
         ifil=ifil+1
      end do
 10   continue
      close(1)
      ire=nint(ifil/2.)
      do j=1,ire
         if((mod(ifil,2).ne.0).and.(j.eq.ire)) then
            write(6,'(i2,3x,a14)')j,name(j)
         else
            write(6,'(i2,3x,a14,10x,i2,3x,a14)')j,name(j),j+ire,
     *         name(j+ire)
         end if
      end do
      call system('rm @datafiles@')
      return
      end
```

```
c
c     ********************** SUBROUTINE PUTIN **********************
c
c     Purpose:  This subroutine puts in all user information into a file
c               for later retrieval.  This allows the user to run a
c               second display for the same profile (with minor
c               changes if so desired) without going
c               through each prompt again.
c
c     ****************************************************************
c
      subroutine putin(rl,ru,irplot,filvar,fun)
      include 'errvar'
      character*3 fun
      open(2,FILE='usin')
      if((erropt.eq.'a').or.(erropt.eq.'A')) then
         filvar=aber
      else
         filvar=per
      end if
      call kyread(7,fun)
      write(2,'(25x,"** USER INPUT PARAMETERS **")')
      write(2,*)
      write(2,'(5x,"You are now in ''vi''.  If there are any changes",
     +" desired use ''vi'' commands")')
      write(2,'(5x,"for editing.  Exit by typing ''ZZ''.")')
      write(2,*)
      write(2,*)
      if(fun.eq.'fps') then
         write(2,'(f7.0,10x,"| Maximum height for display in feet")')
     +   hmax*3.280839
         write(2,'(f4.0,13x,"| Maximum range for display in nm")')
     +   rmax/1.85318
         write(2,'(f6.0,11x,"| Antenna height in feet")')ht*3.280839
      else
         write(2,'(f7.0,10x,"| Maximum height for display in meters")')
     +   hmax
         write(2,'(f4.0,13x,"| Maximum range for display in km")')
     +   rmax
         write(2,'(f6.0,13x,"| Antenna height in meters")')ht
      end if
      write(2,'(f5.0,12x,"| Lower elevation angle (mrad)")')rl
      write(2,'(f5.0,12x,"| Upper elevation angle (mrad)")')ru
      write(2,'(i4,13x,"| Number of rays to be plotted (integer)")')
     +irplot
      write(2,'(a1,16x,"| A or P -- Absolute or Percent error")')erropt
      write(2,'(i1,16x,"| 1 - graph ; 2 - numbers : for profile printout
     + (integer)")')ipro_print
      write(2,*)
      write(2,'(10x,"*** Fill in only the line that applies ***")')
      write(2,*)
      if((erropt.eq.'a').or.(erropt.eq.'A')) then
         if(fun.eq.'fps') then
            write(2,'(f5.0,12x,"| Error increment in feet")')
     +      filvar*3.280839
         else
```

```
            write(2,'(f5.0,12x,"| Error increment in meters")')filvar
         end if
         write(2,'(17x,"| Percent error increment")')
      else if((erropt.eq.'p').or.(erropt.eq.'P')) then
         write(2,'(17x,"| Error increment")')
         write(2,'(f3.0,14x,"| Percent error increment")')filvar
      end if
      close(2)
      call rdffl(rl,ru,irplot,filvar,fun)
      return
      end
c
c     ********************** SUBROUTINE RDFFL ************************
c
c     Purpose:  This subroutine reads all user input parameters from a
c               file.
c
c     ****************************************************************
c
      subroutine rdffl(rl,ru,irplot,filvar,fun)
      include 'errvar'
      character*3 fun
      character*80 dum
      call system('vi usin')
      open(2,FILE = 'usin')
      read(2,'(5(a80/),a80)')(dum,i=1,6)
      read(2,'((f7.0/),(f4.0/),f6.0)')hmax,rmax,ht
      read(2,'(2(f5.0/),(i4/),(a1/),i1)')rl,ru,irplot,erropt,
     +ipro_print
      if(fun.eq.'fps')then
         hmax=hmax/3.280839
         rmax=rmax*1.85318
         ht=ht/3.280839
      end if
      read(2,'(2(a80/),a80)')(dum,i=1,3)
      if((erropt.eq.'a').or.(erropt.eq.'A')) then
         read(2,'(f5.0)')filvar
         aber=filvar
         if(fun.eq.'fps') aber=aber/3.280839
      else if((erropt.eq.'p').or.(erropt.eq.'P')) then
         read(2,'((a80/),f5.0)')dum,filvar
         per=filvar
      end if
      close(2)
      do j=1,35
         h(j)=0.
         m(j)=0.
         dmdh(j)=0.
      end do
      do j=1,300
         angle(j)=0.
      end do
      return
      end
```

```
c  ********************* SUBROUTINE READDA ****************************
c
c     Purpose:  Subroutine READDA reads environmental data from data
c               files the user specifies.
c
c     Glossary:
c
c        profile     Character array containing environmental data
c                    filenames.
c        ipick       Counter for "profile" array indicating number
c                    of environmental data file.
c
c  *******************************************************************
c
      subroutine readda(ipick,profile)
$INCLUDE 'errvar'
      character*14 profile(50)
      character*80 dum
c
c    ****************** Open data file; read data *********************
c
      open(10,FILE='.prof/'//profile(ipick))
      read(10,'(3(a80/),a80)')(dum,i=1,4)
      read(10,'((a14/),(a1/),a1)')profile(ipick),hun,reun
      read(10,'(4(a80/),a80)')(dum,i=1,5)
      ilvl=0
c
c  ********************* Read H and M arrays. ***********************
c
      do i=1,33
         if(reun.eq.'M') then
            read(10,'(f7.1,5x,f6.1)',END=10)orh(i),orm(i)
         else
            read(10,'(f7.1,18x,f6.1)',END=10)orh(i),orm(i)
         end if
         ilvl=ilvl+1
      end do
 10   continue
      close(10)
      return
      end
```

```
c  ************************* SUBROUTINE REGULAR **********************
c
c     Purpose:  This begins the main calculations of the program.  It
c               does not allow rays to be traced within an angular limit
c               if the transmitter height is at a break point.
c
c     Glossary:
c
c         duct     Logical flag indicating if the transmitter height is
c                  at a break point.
c         irplot   Number of rays to be plotted.
c         alpha    Beginning angle.
c         fixang   Initial elevation angle currently being used.
c         ij       Counter.
c         hbef     Beginning height.
c         rbef     Beginning range.
c         forbid   Rays are not traced in this angular range.
c
c   *****************************************************************
c
      subroutine regular(irays,duct,forbid,ix)
      include '/usr/include/starbase.f2.h'
      include '/usr/include/starbase.f1.h'
$INCLUDE 'errvar'
      logical duct
c
c  **************** Begin main loop. *********************************
c
      do ik=1,irays
         frst=.true.
         if(plot)frstime=.true.
         if(im.ne.1) then
            mvfl=.false.
         else
            if(plot) then
               mvfl=.true.
            else
               if(f7.eq.'fps') then
                  call move2d(fildes,0.,ht*3.280839)
               else
                  call move2d(fildes,0.,ht)
               end if
            end if
         end if
c
c   *************** Initialize variables for ray tracing. *************
c
         if(plot) call start(ik,ix)
         alpha=angle(ik)
         fixang=angle(ik)
         ij=ix
         hbef=ht
         rbef=0.
c
c  *********** Check if transmitter is at a breakpoint.  If so, skip
c  angles within forbidden range. *********************************
```

```
c
            if(duct) then
               if(ABS(alpha).le.forbid) then
                  rp=rmax
                  call line_color_index(fildes,1)
                  if(f7.eq.'fps') then
                     call draw2d(fildes,rp*.5396117,ht*3.280839)
                  else
                     call draw2d(fildes,rp,ht)
                  end if
                  goto 50
               end if
            end if
c
c  *********** If initial elevation angle is negative then call routine
c  for down-going rays.  If it is positive, then call routine for up-
c  going rays.  If it is 0  then call routine to check on the value of
c  the gradient. ***********************************************************
c
 30         if(alpha.eq.0.) then
               call zero(alpha,hbef,ij,rbef,rp,hp,fixang)
            else if(alpha.gt.0.) then
               call up(alpha,hbef,ij,rbef rp,hp,fixang)
            else
               call down(alpha,hbef,ij,rbef,rp,hp,fixang)
            end if
            if(hp.eq.h(1)) goto 50
            if((rp.lt.rmax).and.(hp.lt.hmax)) goto 30
 50         continue
         end do
         return
         end
c
c        ************************ SUBROUTINE SEVEN ***********************
c
c        Purpose:  This routine performs the ray-tracing procedure 7 times
c                  (once for each color) for ease in plotting onto the
c                  graphics plotter.
c
c        ****************************************************************
c
         subroutine seven(irays,duct,forbid,ix)
$INCLUDE 'errvar'
         logical duct
         do ia=1,7
            mvfl=.true.
            call regular(irays,duct,forbid,ix)
         end do
         return
         end
```

```
c     ********************** SUBROUTINE STEPFIXSCR **********************
c
c     Purpose:  STEPFIXSCR defines the color scale for plotting height
c               and range onto the screen in the correct color.
c
c     Glossary:
c
c         percerr Percentage error between height of non-standard
c                 atmosphere ray and height of standard atmosphere
c                 ray.
c         rng     Interpolated range between color transitions.
c         hyt     Interpolated height between color transitions.
c
c     *******************************************************************
c
      subroutine stepfixscr(hdif,hs,rp,hp,rng,hyt)
      include '/usr/include/starbase.f1.h'
      include '/usr/include/starbase.f2.h'
$INCLUDE 'errvar'
      if(rng.ne.0.) then
         if(f7.eq.'fps') then
            call draw2d(fildes,rng/1.85318,hyt*3.280839)
         else
            call draw2d(fildes,rng,hyt)
         end if
      end if
c
c     ************ Define color scale for absolute height error. *********
c
      if((erropt.eq.'a').or.(erropt.eq.'A')) then
         if(hdif.le.aber) then
            call line_color_index(fildes,1)
         else if((hdif.gt.aber).and.(hdif.le.2.*aber)) then
            call line_color_index(fildes,3)
         else if((hdif.gt.2.*aber).and.(hdif.le.3.*aber)) then
            call line_color_index(fildes,4)
         else if((hdif.gt.3.*aber).and.(hdif.le.4.*aber)) then
            call line_color_index(fildes,5)
         else if((hdif.gt.4.*aber).and.(hdif.le.5.*aber)) then
            call line_color_index(fildes,6)
         else if((hdif.gt.5.*aber).and.(hdif.le.6.*aber)) then
            call line_color_index(fildes,7)
         else if(hdif.gt.6.*aber) then
            call line_color_index(fildes,2)
         end if
c
c    *********** Define color scale for percentage height error. ********
c
      else if((erropt.eq.'p').or.(erropt.eq.'P')) then
         if(hs.eq.1.) then
            percerr=200.
         else
            percerr=(hdif/hs)*100.
         end if
         if(percerr.le.per) then
            call line_color_index(fildes,1)
```

```
         else if((percerr.gt.per).and.(percerr.le.2.*per)) then
            call line_color_index(fildes,3)
         else if((percerr.gt.2.*per).and.(percerr.le.3.*per)) then
            call line_color_index(fildes,4)
         else if((percerr.gt.3.*per).and.(percerr.le.4.*per)) then
            call line_color_index(fildes,5)
         else if((percerr.gt.4.*per).and.(percerr.le.5.*per)) then
            call line_color_index(fildes,6)
         else if((percerr.gt.5.*per).and.(percerr.le.6.*per)) then
            call line_color_index(fildes,7)
         else if(percerr.gt.6.*por) then
            call line_color_index(fildes,2)
         end if
      end if
      if(f7.eq.'fps') then
         call draw2d(fildes,rp/1.85318,hp*3.280839)
      else
         call draw2d(fildes,rp,hp)
      end if
      return
      end
c
c     ********************** SUBROUTINE RENEW ***************************
c
c     Purpose:  RENEW re-initializes arrays used in ray tracing.
c
c    ********************************************************************
c
      subroutine renew
$INCLUDE 'errvar'
      do j=1,35
         dmdh(j)=0.
         h(j)=0.
         m(j)=0.
      end do
      do j=1,33
         orh(j)=0.
         orm(j)=0.
      end do
      do j=1,300
         angle(j)=0.
      end do
      return
      end
```

```
c  ************************* SUBROUTINE UP ****************************
c
c     Purpose:  This subroutine performs a ray trace for up-going rays.
c
c  ********************************************************************
c
      subroutine up(alpha,hbef,ij,rbef,rp,hp,fixang)
      include '/usr/include/starbase.f1.h'
      include '/usr/include/starbase.f2.h'
$INCLUDE 'errvar'
c
c  ********** Begin at transmitter height and calculate until the ray
c  reaches a maximum or maximum height. ****************************
c
      do while((ij.lt.nlvl).and.(alpha.ge.0.))
         rp=rbef+rinc
         if(rp.ge.rmax) goto 20
         alphap=alpha+dmdh(ij)*(rp-rbef)
c
c  ********* If the ray reaches a maximum then set ALPHAP to 0. and
c  calculate height and range. Call DOWN. *************************
c
         if(alphap.le.0.) then
            alphap=0.
            rp=rbef-alpha/dmdh(ij)
            hp=hbef-alpha**2/2.e-3/dmdh(ij)
         end if
         hp=hbef+(alphap**2-alpha**2)/2.e-3/dmdh(ij)
         if(hp.gt.h(ij+1)) then
            hp=h(ij+1)
            rad=alpha**2+2.e-3*dmdh(ij)*(hp-hbef)
c
c  ********* If the ray reaches a maximum then set ALPHAP to 0. and
c  calculate height and range. Call DOWN. ************************
c
            if(rad.le.0.) then
               alphap=0.
               rp=rbef-alpha/dmdh(ij)
               hp=hbef-alpha**2/2.e-3/dmdh(ij)
            else
               alphap=SQRT(rad)
               rp=rbef+(alphap-alpha)/dmdh(ij)
            end if
            ij=ij+1
         end if
         if((rp.ge.rmax).or.(hp.ge.hmax)) goto 10
c
c   ********** Call STAND to determine height of ray for standard
c   atmosphere at same range. **********************************
c
         call stand(fixang,rp,hp)
         hbef=hp
         rbef=rp
         alpha=alphap
         if(alpha.eq.0.) goto 10
      end do
```

```
 10   if(ij.eq.nlvl) ij=nlvl-1
c
c  *********** If ray is calculated past the maximum height, then set
c  HP equal to HMAX and calculate range at exactly the maximum height. *
c
      if(hp.ge.hmax) then
         hp=hmax
         alphap=SQRT(alpha**2 + 2.e-3*dmdh(ij)*(hp-hbef))
         rp=rbef+(alphap-alpha)/dmdh(ij)
         if(rp.ge.rmax) goto 20
         call stand(fixang,rp,hp)
      end if
c
c   ********** If ray reaches beyond maximum range then calculate height
c   at exactly maximum range. *****************************************
c
 20   if(rp.ge.rmax) then
         rp=rmax
         alphap=alpha+dmdh(ij)*(rp-rbef)
         hp=hbef+(alphap**2-alpha**2)/2.e-3/dmdh(ij)
         call stand(fixang,rp,hp)
      end if
      return
      end
```

```
c  *********************** SUBROUTINE ZERO *****************************
c
c     Purpose:  This subroutine determines which subroutine will be
c               called when a ray reaches a maximum or minimum.  If
c               a ray reaches a maximum, then depending on the value
c               of the gradient, DOWN is called.  Likewise, if a ray
c               reaches a minimum, UP is called.
c
c  *******************************************************************
c
      subroutine zero(alpha,hbef,ij,rbef,rp,hp,fixang)
$INCLUDE 'errvar'
      if(dmdh(ij).lt.0.) then
         call down(alpha,hbef,ij,rbef,rp,hp,fixang)
      else
         call up(alpha,hbef,ij,rbef,rp,hp,fixang)
      end if
      return
      end
```